TODD GARNER

DS 6306 WEEK 6 PART 1

FEBRUARY 87. 2023

READ IN THE DATA AND CLEAN – PART 1, #1

- The site www.opendatasoft.com proved challenging so I found the .csv files on kaggle.com and also in the class GitHub repo.

~~~~
```{r}
# Load the Titanic dataset
Titanic <- read.csv(file.choose(), header = TRUE)
library(dplyr)
Titanic_new <- Titanic %>% select(Survived, Pclass, Age)
Titanic_exp <- Titanic %>% select(Pclass, Age)
Titanic_new_final <- Titanic_new[complete.cases(Titanic_new), ]
Titanic_final <- Titanic_exp[complete.cases(Titanic_exp), ]
```
~~~~

- Two data sets: training and test. Removed the NA's and selected the three pertinent columns.
- Comment: I used ChatGPT and found it distracted me more than it assisted me. I may use it in a different manner than writing the code base.

CREATE THE TRAINING/TESTING SETS, USE THEM AND CONFUSION MATRIX – PART 1, #2

```r
set.seed(123)
train_new_index = sample(1:nrow(Titanic_new_final), size = nrow(Titanic_new_final)*0.8, replace = FALSE)
train_new = Titanic_new_final[train_new_index, ]
test_new = Titanic_new_final[-train_new_index, ]

set.seed(123)
train_index = sample(1:nrow(Titanic_final), size = nrow(Titanic_final)*0.8, replace = FALSE)
train = Titanic_final[train_index, ]
test = Titanic_final[-train_index, ]

library(class)
library(caret)
knn.23 <- knn(train, test, train_new$Survived, k = 23)
knn.24 <- knn(train, test, train_new$Survived, k = 24)
table(test_new$Survived, knn.23)
confusionMatrix(table(test_new$Survived, knn.23))

table(test_new$Survived, knn.24)
confusionMatrix(table(test_new$Survived, knn.24))
```

- Data set: training -> 571 testing -> 143 (177 NA's removed)
- Transforming in KNN and Confusion Matrix (results on next slide)

CONFUSION MATRIX AND STATISTICS

```
Confusion Matrix and Statistics

   knn.24
     0  1
  0 69 10
  1 46 18

               Accuracy : 0.6084
                 95% CI : (0.5233, 0.6889)
    No Information Rate : 0.8042
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1634

 Mcnemar's Test P-Value : 2.91e-06

            Sensitivity : 0.6000
            Specificity : 0.6429
         Pos Pred Value : 0.8734
         Neg Pred Value : 0.2813
             Prevalence : 0.8042
         Detection Rate : 0.4825
   Detection Prevalence : 0.5524
      Balanced Accuracy : 0.6214

       'Positive' Class : 0
```

Having seen the answers, I know now that my solution was off the mark. I'm impressed with your code. Specific and on target. I have much to learn.

AGE AND PROBABILITY OF SURVIVAL, PART 1, #3

~~~~
```{r}
knn.23 <- knn(train, test, train_new$Survived, k = 23)
knn.24 <- knn(train, test, train_new$Survived, k = 24)

table(test_new$Survived, knn.23)
confusionMatrix(table(test_new$Survived, knn.23))
table(test_new$Survived, knn.24)
confusionMatrix(table(test_new$Survived, knn.24))
test_new <- data.frame(Pclass = 1, Age = 59)
test_new.2 <- data.frame(Pclass = 2, Age = 59)
test_new.3 <- data.frame(Pclass = 3, Age = 59)
knn.todd_1 <- knn(train, test_new, train_new$Survived, k = 24)
knn.todd_2 <- knn(train, test_new.2, train_new$Survived, k = 24)
knn.todd_3 <- knn(train, test_new.3, train_new$Survived, k = 24)
~~~~

Through my research, I found one suggestion on the value to select for k. It was suggested that k can be derived from the square root of the size of the data set.

The result is between k = 23 and k = 24. I've employed them both

I created 3 data sets with my Age (59) in it along with the Pclass. 1st, 2nd, 3rd.

```
[1] 0
Levels: 0 1
[1] 0
Levels: 0 1
[1] 0
Levels: 0 1
```

Given the data, it looks bad for someone my age, regardless of the class.

USE OF THE TRAINING SET AND THE TEST SET – PART 1 #4

- Utilizing the prior training set (571 observations), I've loaded in the test_set.csv. This is where I got stuck and moved to Part 2.

~~~~
```{r}
Titanic_418 <- read.csv(file.choose(), header = TRUE)
Titanic_418_new <- Titanic_418 %>% select(Pclass, Age)
Titanic_418_final <- Titanic_418_new[complete.cases(Titanic_418_new), ]
knn.418_final <- knn(train, Titanic_418_final, train_new$Survived, k = 18)
train_new$Survived <- as.factor(train_new$Survived)
```
~~~~

PART 2 # A. IRIS DATA SET – 70/30 TRAINING SET, PLOT XAXIS

```
```{r}
library(tidyverse)
iris
view(iris)
splitPerc = .70
train_init = sample(1:dim(iris)[1],round(splitPerc * dim(iris)[1]))
train = iris[train_init,]
test = iris[-train_init,]
```
```

Description: df [150 x 5]

| Sepal.Length <dbl> | Sepal.Width <dbl> | Petal.Length <dbl> | Petal.Width <dbl> | Species <fctr> |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |

# 70/30 TRAIN/TEST | ITERATE ALL K VALUES

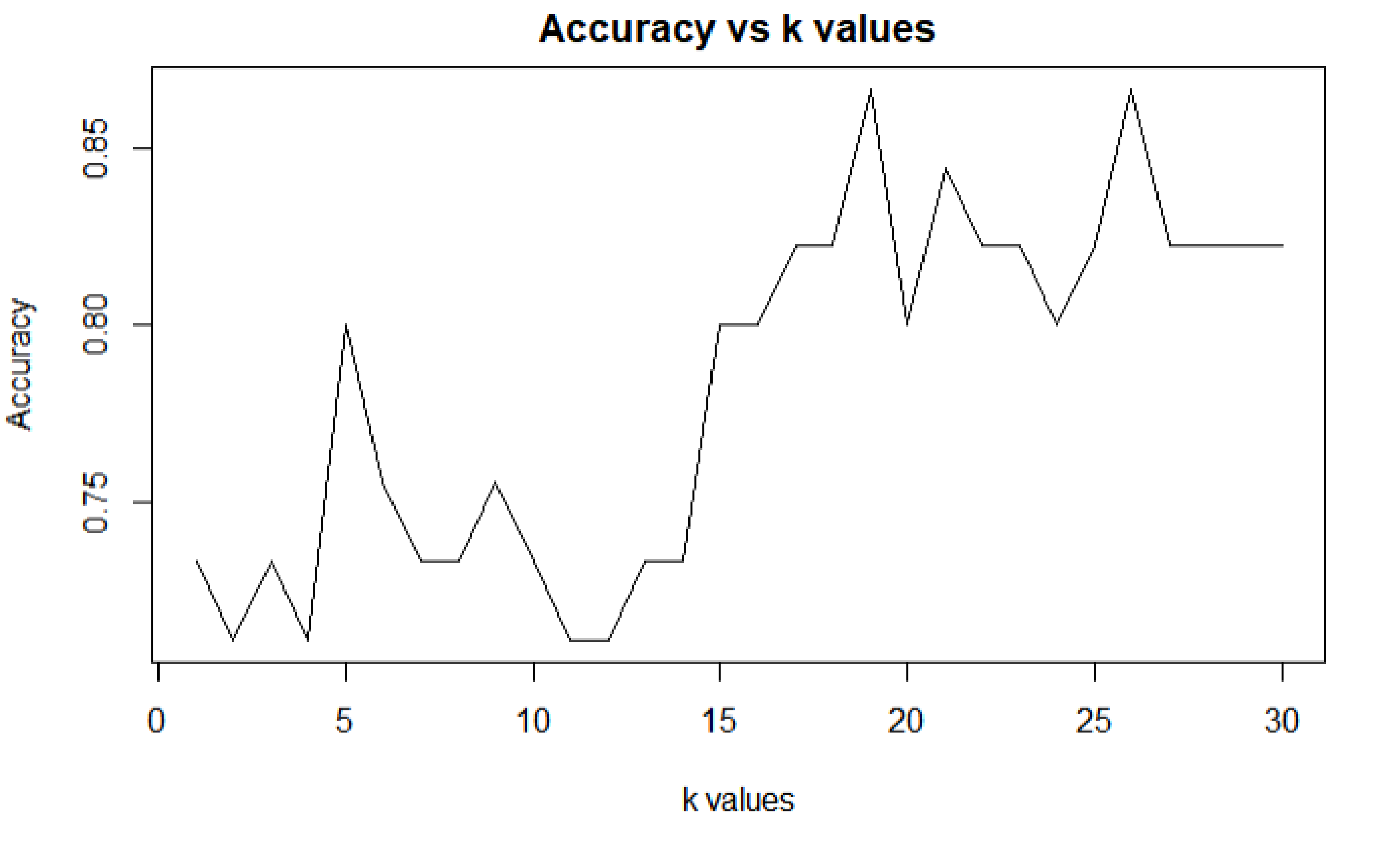
```
iris
library(class)
library(FNN)
library(caret)
library(ISLR)
set.seed(6)
splitPerc = .70
trainIndices = sample(1:dim(iris)[1],round(splitPerc * dim(iris)[1]))
train = iris[trainIndices,]
test = iris[-trainIndices,]
iris %>% ggplot(aes(x = Sepal.Length,Sepal.Width,color = Species)) + geom_point()
View(train$Species)
# k = 5
classifications = knn(train[,c(1,2)],test[,c(1,2)],train$Species, prob = TRUE, k = 5)
table(test$Species,classifications)
confusionMatrix(table(test$Species,classifications))
```

Iris – Split the data into 70/30 train/test sets, plot the results (plots on the following slide), classify and employ the confusion matrix.

```
accs = data.frame(accuracy = numeric(30), k = numeric(30))
nrow(train)
for(i in 1:30)
{
classifications = knn(train[,c(1,2)],test[,c(1,2)],train$Species, prob = TRUE, k = i)
table(test$Species,classifications)
CM = confusionMatrix(table(test$Species,classifications))
accs$accuracy[i] = CM$overall[1]
accs$k[i] = i
}
plot(accs$k,accs$accuracy, type = "l")
```

Create a "for loop" to test the k values to find the best possible fit.

DETERMINATION OF BEST K VALUE | ANSWER = 18



CONFUSION MATRIX RESULTS

```
  setosa          13          0          0
  versicolor       1         11          4
  virginica        0          3         13

Overall Statistics

               Accuracy : 0.8222
                 95% CI : (0.6795, 0.92)
    No Information Rate : 0.3778
    P-Value [Acc > NIR] : 1.262e-09

                  Kappa : 0.7327

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 0.9286            0.7857           0.7647
Specificity                 1.0000            0.8387           0.8929
Pos Pred Value              1.0000            0.6875           0.8125
Neg Pred Value              0.9688            0.8966           0.8621
Prevalence                  0.3111            0.3111           0.3778
Detection Rate              0.2889            0.2444           0.2889
Detection Prevalence        0.2889            0.3556           0.3556
Balanced Accuracy           0.9643            0.8122           0.8288
[1] 105
```

BONUS QUESTION: REPEAT PRIOR ANALYSIS WITH LOOCV METHODOLOGY - RESULTS

```
     setosa         50         0        0
     versicolor      0        31       12
     virginica       0        19       38

Overall Statistics

               Accuracy : 0.7933
                 95% CI : (0.7197, 0.8551)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.69

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.6200           0.7600
Specificity                 1.0000            0.8800           0.8100
Pos Pred Value              1.0000            0.7209           0.6667
Neg Pred Value              1.0000            0.8224           0.8710
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.2067           0.2533
Detection Prevalence        0.3333            0.2867           0.3800
Balanced Accuracy           1.0000            0.7500           0.7850
[1] 105
```

These results are lower slightly than the k = 18 confusion matrix results on the prior page.

I think this highlights that LOOCV is great to use on small-ish data sets. On data sets with hundreds of values/observations, it appears that LOOCV is not superior to determining an external (XX%/XX%) versus an internal (LOOCV) methodology.